ET420197585US

10042103.010702

FIXED SNOOP RESPONSE TIME FOR SOURCE-CLOCKED MULTIPROCESSOR BUSSES

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates generally to a multiprocessor system and, more particularly, to maintaining identical bus delays for different processors in a multiprocessor system.

Description of the Related Art

In a switch-based multiprocessor system using high-speed, source-clocked, unidirectional point-to-point busses, with different wiring delay timing differences between busses, the natural choice for implementation of a snoop based protocol, would be to allow variations in snoop address and snoop response times. However, this introduces complexity in the design.

In a standard snoop protocol used in a signal bus, bus masters, which take control of the bus, arbitrate for the bus and present their address and command on the bus when access is granted. Each processor or memory controller attached to the bus sees the address and command at the same time and generates its snoop response at a time specified by the bus architecture. Then, the snoop response becomes valid after the snoop request has been received by the snooping memory controller and processor.

In a multiprocessor system, two or more processors source commands and addresses on processor outbound busses to a memory controller. Typically, the memory controller may function as a bus switch and an address switch. The memory controller arbitrates between the processor busses, selecting one processor outbound command to reflect back to all the processors, via processor inbound busses. Since there may be wiring delay differences between the processor inbound busses, a command provided to the processors at the same memory controller clock may not arrive at the processors at the same time. Similarly, when the multiprocessor system has point-to-point, unidirectional, source-clocked snoop response busses, these busses carrying snoop responses may also have wiring delay differences between the processors.

The differences in bus delays complicate the snoop protocol, if differences are allowed between when each processor observes the snoop response for a particular snoop. In addition, the memory controller job of combining the snoop responses is more difficult, if the memory controller sees the responses for a particular snoop at different times from each processor.

Therefore, there is a need for aligning the snoop addresses and snoop responses across all busses. This would allow the snoop protocol to be a simple variant of single bus based snoop protocol.

SUMMARY OF THE INVENTION

The present invention provides a multiprocessor system.

In one embodiment of the present invention, a first microprocessor has one or more interfacing logics including a first interfacing logic. The first microprocessor is clocked by a first system clock. A memory controller is connected to the first interfacing logic through at least a first bus for transmitting at least a first signal from the memory controller to the first interfacing logic. The memory controller is clocked by a second system clock. A second microprocessor is connected to the memory controller through at least a second bus for transmitting at least a second signal from the memory controller to the second processor. The second bus requires a first period of time more to transmit the second signal than what the first bus requires to transmit the first signal. The first interfacing logic delays the first signal by the first period of time so that the first and the second signals are respectively received by the first and the second microprocessors substantially at the same time.

In another embodiment of the present invention, a memory controller has one or more interfacing logics including a first interfacing logic. The memory controller is clocked by a first system clock. A first microprocessor is connected to the first interfacing logic through at least a first bus for transmitting at least a first signal from the first microprocessor to the first interfacing logic. The first microprocessor is clocked by a second system clock. A second microprocessor is connected to the memory controller through at least a second bus for transmitting at least a second signal from the second processor to the memory controller. The second bus requires a first period of time more to transmit the second signal than what the first bus requires to transmit the first signal. The first interfacing logic delays the first signal by the first period of time so that the first and the second signals are respectively received by the first and the second microprocessors substantially at the same time.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

FIGURE 1 depicts a multiprocessor system and a bus configuration thereof;

FIGURE 2 depicts a block diagram showing one embodiment of an interfacing logic connected to a bus of the multiprocessor system as shown in FIGURE 1; and

FIGURE 3 depicts a timing diagram showing control signals used in FIGURE 2 in an embodiment of the present invention.

DETAILED DESCRIPTION

The principles of the present invention and their advantages are best understood by referring to the illustrated operations of embodiment depicted in FIGURES 1-3.

In FIGURE 1, a reference numeral 100 designates a multiprocessor system having four processors 102, 104, 106, and 108, each of which is connected to a memory controller 110. The processors 102, 104, 106, and 108 each represents any type of processor having computing capabilities. Also, the number of processors may vary depending on the configuration of the multiprocessor system 100. The memory controller 110 has address and bus switch functionalities. Alternatively, the memory controller 110 is replaceable with address switch without departing from the true spirit of the present invention.

The processor 102 is connected to the memory controller 110 through an address/data outbound bus 112 for transmitting addresses and data from the processor 102 to the memory controller 110. An address/data inbound bus 114 is also shown to connect the processor 102 and the memory controller 110 for transmitting addresses and data from the memory controller 110 to the processor 102. A snoop response outbound bus 116 is shown to connect the processor 102 and the memory controller 110 for transmitting snoop responses from the processor 102 to the memory controller 110. A snoop response inbound bus 118 is shown to connect the processor 102 and the memory controller 110 for transmitting snoop responses from the memory controller 110 to the processor 102.

The other three processors 104, 106, and 108 are connected to the memory controller in a similar fashion. An address/data outbound bus 120, an address/data inbound bus 122, a snoop response outbound bus 124, and a snoop response inbound bus 126 are similarly shown to connect the processor 104 and the memory controller 110. Likewise, an address/data outbound bus 128, an address/data inbound bus 130, a snoop response outbound bus 132, and a snoop response inbound bus 134 are shown to connect the processor 106 and the memory controller 110. Finally, an address/data outbound bus 136, an address/data inbound bus 138, a snoop response outbound bus 140, and a snoop response inbound bus 142 are shown to connect the processor 108 and the memory controller 110.

The multiprocessor system 100 preferably uses a high frequency (e.g., 1 GHz), point-to-point, unidirectional, source-clocked busses. The processors 102, 104, 106, and 108 source addresses and commands (i.e., data) on their respective address/data outbound busses 112, 120, 128, and 136 to the memory controller 110. As mentioned above, the memory controller 110 implements a system bus switch. Thus, the memory controller 110 arbitrates between the four processor busses, selecting one processor outbound command to reflect back to all four processors 102, 104, 106, and 108, via their respective address/data inbound busses 114, 122, 130, and 138. Since there may be wiring delay differences between the four processor inbound busses 114, 122, 130, and 138, a command sourced to the processor at a memory controller clock (not shown) may not arrive at the processors 102, 104, 106, and 108 at the same time.

Similarly, the multiprocessor system 100 has point-to-point, unidirectional, source-clocked snoop response busses. The snoop response outbound busses 116, 124, 132, and 140 carry the snoop responses of the respective processors 102, 104, 106, and 108. The snoop response inbound busses 118, 126, 134, and 142 carry a snoop response, which is a combination by the memory controller 110 of the snoop responses of all the processors 102, 104, 106, and 108. These snoop response busses may also have wiring delay differences between the processors 102, 104, 106, and 108.

A reference numeral 144 designates an interfacing logic at the receiving end of each of the busses 112 through 142 as shown in FIGURE 1. Preferably, the interfacing logic 144 is implemented in the processors 102, 104, 106, and 108, as well as in the memory controller 110. The interfacing logics 144 implemented in the processors enable all the processors 102, 104, 106, and 108 to receive their snoop commands or snoop responses at the same bus clock by adding delay to busses with less delay to remove any delay differences between the busses directed to the processors 102, 104, 106, and 108. Likewise, the interfacing logics 144 implemented in the memory controller 110 enable the memory controller 110 to receive all snoop responses at the same bus clock by adding delay to busses with less delay to remove any delay differences between the busses directed to the memory controller 110.

Referring now to FIGURE 2, one embodiment of the interfacing logic 144 is shown to be connected to a data bus 200. The data bus 200 can be any of the busses 112 through 142 as shown in FIGURE 1. As mentioned above, the interfacing logic 144 is implemented in the receiving end of the data bus 200. The data carried on the data bus 200 can be any of a snoop command, a snoop response, an address, and a command, depending on the type of the data bus 200. Generally, the data bus 200 is n-bit wide for data transmission, and m-bit wide for clock transmission (n and m are integers larger than zero).

The interfacing logic 144 has a chip receiver 202 for receiving the data transmitted on the data bus 200. Optionally, the chip receiver 202 is connected to a deskew circuit 204. The deskew circuit 204 generally comprises a delay mechanism for adjusting delay differences between different bit lines in the data bus 200. Since different bit lines in the same data bus may lead to different delays, the deskew circuit 204 compensates for the difference. The data bus 200 also transmits a bus clock bus_clk from a launch chip (not shown). The launch chip is implemented either in the memory clock 110 of FIGURE 1 or in one of the processors 102, 104, 106, and 108 of FIGURE 1, depending on the location of the interfacing logic 144 in FIGURE 1. In either case, the bus_clk is the same as, or derived from, a bus clock (not shown) of the launch chip. For example, if the data bus 200 represents the address/data outbound bus 112 of FIGURE 1, then the bus_clk is the same as, or derived from, a bus clock of the processor 102 of FIGURE 1. If the data bus 200 represents the address/data inbound bus 114 of FIGURE 1, then the bus_clk is the same as, or derived from, a bus clock of the memory controller 110.

A chip receiver 206 is connected to the data bus 200 for receiving the bus_clk. The chip receiver 206 is also connected to a deskew circuit 208. The deskew circuit 208 adjust delay differences between different bit lines. Additionally, the deskew circuit 208 does the job of splitting the bus_clk into c1-c4 clock signals. Alternatively, a clock generator (now shown) could be used to split the bus_clk into c1-c4 clock signals. Preferably, the c1 and c3 clock signals are the deskewed version of the bus_clk. The c2 and c4 clock signals are the inversions of the c1 and c3 clock signals, respectively.

The deskew circuit 204 is connected to four select circuits 210, 212, 214, and 216 for sending data to the four select circuits 210, 212, 214, and 216. The select circuits 210, 212, 214, and 216 are connected to latches 218, 220, 222, and 224, respectively, for sending data to the respective latches 218, 220, 222, and 224, and for receiving feedback data from the respective latches 218, 220, 222, and 224. The select circuits 210, 212, 214, and 216 are controlled by control signals g1, g2, g3, and g4, respectively. The select circuits 210, 212, 214, and 216 are configured to output the data received from the deskew circuit 201 when the control signals are asserted, and are configured to output the feedback data received from the latches 218, 220, 222, and 224 when the control signals are deasserted. The deskew circuit 208 is connected to the latches 218, 220, 222, and 224 for clocking them using the c1, c2, c3, and c4 signals, respectively. As mentioned above, the c1, c2, c3, and c4 signals are derived from the bus_clk. The latches 218, 220, 222, and 224 each may be replaced with a register (not shown) comprising a N number of latches (not shown). In that case, the data received by the interfacing circuit 144 is N bits.

A multiplexer 226 is connected to the latches 218, 220, 222, and 224 for receiving data d1, d2, d3, and d4, respectively. The multiplexer 226 is also connected to a latch 228 for outputting data. A control signal g5 controls the multiplexer 226. The control signals g1, g2, g3, g4, and g5 received by the multiplexers 210, 212, 214, 216, and 226, respectively, are derived from a control logic (now shown) implemented in the receiving end of the data transmission. The latch 228 is also connected to a clock distributor clk_dist 230 for receiving system clock sys_clk signal. The sys_clk signal is a system clock signal of the receiving end. The clk_dist is connected to a chip receiver 232. The latch 228 outputs a data_out signal and receives a new data at the rising edge of the clock signal c5. Optionally, a clock generator (not shown) may be inserted between the chip receiver 232 and the clk_dist 230 for generating the c5 clock having different frequency from that of the sys_clk signal.

Preferably, the control signals g1 through g4 are generated by a first local logic (not shown), which is driven by the deskewed bus_clk. The control signal g5 is generated by a second local logic (not shown), which is driven by the c5 clock. A detailed sequence of the control signals g1 through g5 is shown in FIGURE 3. The sequence of the control signal g5 relative to the receiving data d1, d2, d3, and d4 can be changed by a programmable parameter (not shown) such that the data_out signal coming from the latch 228 is delayed by a variable amount relative to the data out of the deskew circuit 204. Preferably, the programmable parameter contains information on the amount of such delay.

The data bus 200 represents any one of sixteen busses shown in FIGURE 1. Assume here that the data bus 200 is one of four address/data inbound busses as shown in FIGURE 1. These busses are used to transmit data from the memory controller 110 to the processors 102, 104, 106, and 108. When the bus_clk is operating at a high frequency such as 1 GHz, the data transmit time through the data bus 200 may be greater than one bus_clk. Many factors such as bus and data control logic, chip placement, and interchip wiring rules, and bus physical layer controls the skew between the bits on a single bus. Therefore, delays of different busses may be different. These delay differences are handled by delaying the transferring of data to the local latch on faster busses by one or more bus clocks. FIGURE 2 shows a circuit configuration of a receiving end of a bus with more delay than other busses. The interfacing logic 144 is configured for delaying up to three bus clocks.

The number of select circuits and latches is changeable without departing from the true spirit of the present invention. Here, four select circuits and four latches are used. It can be any plural number, depending on how much delay is necessary.

In FIGURE 3, a timing diagram 300 depicts the clock signals and the control signals as shown in FIGURE 2. The timing diagram 300 presents the operation of the interfacing logic 144. The data as noted in the timing diagram 300 represents the data output from the deskew 204 of FIGURE 2. Before time t0, data a is input to the select circuits 210, 212, 214, and 216. Right before time t0, the control signals g1 and g4 were asserted. Thus, the latches 218 and 224 received the data from the select circuits 210 and 216, respectively. At time t0, the clock signals c1 and c3 are deasserted, whereas the clock signals c2 and c4 are asserted. Assuming that the latches are triggered at the rising edges of clock pulses, the multiplexer 226 will receive updated inputs from the latches 220 and 224 at time t0. Since the multiplexer 212 outputs the feedback data received from the latch 220, however, the multiplexer 226 will only receive a new data d4 from the latch 224 at time t0. Thus, the data a block is shown in d4 after time t0. The data a passes through the multiplexer 226 when the g5 control signal selects the output of the multiplexer 224. This is shown in the timing diagram 300 as between time t0 and t1. It is noted here that the g5 control signal did not select the output of the multiplexer 224 when the data a is input the select circuits 216. Rather, the g5 control signal intentionally delayed this action by one clock cycle of the sys_clk signal. After the rising edge of the g5 signal for selecting the data a, the latch 228 outputs the data a at the rising edge of the c5 clock signal, which occurs at time t1. Thus, the data a is carried in the data_out signal slightly after the time t1 as shown in the timing diagram 300.

Similarly, subsequent data b, c, d, e, f, and g are transmitted through the data bus 200 and go through the interfacing logic 144 before carried over to a local logic (not shown) of receiving end. Therefore, all data transmitted through the data bus 200 would be delayed by one cycle of the c5 signal. The amount of delay in the number of clock cycle of the *c5* signal is determined by the bus control logic generating the control signals g1-5.

It will be understood from the foregoing description that various modifications and changes may be made in the preferred embodiment of the present invention without departing from its true spirit. This description is intended for purposes of illustration only and should not be construed in a limiting sense. The scope of this invention should be limited only by the language of the following claims.